Code of honor

Copy and sign the following statement into the first page of the exam resolution

By submitting this assessment online, I declare on my honor that I will solve the test using only the authorized consultation elements, autonomously and without exchanging any information or communicating by any means, with any person or information repository, physical or virtual.

This exam has 3 pages. The last page contains one question and information about the resolution submission.

Systems [2.0]

When defining a complex system using a systemic approach, one of the principles is that a system can be broken into a set of smaller subsystems.

1. [2.0 points]

- Present two advantages of the application of this principle when developing a system
- Apply this principle to the project and present and describe the responsibility of three subsystems that composed it.
- Describe how these subsystems would interact with each other and the environment.

1.0	It is simpler to develop each os the subsystems individualty It is easier to find the errors
0.5	Communicaction sub-sistems responsib;e for all the connections and message formating Game rules responsible for the verification of the rules of the game
0.5	The communication module would forward all messages to the rules

Software architectures [4.0]

2. [1.0 points]

- Present two architectural patterns that help the programmer organize the code base.
- Describe both, and present the main differences between them.

0.5	Component-Based Object-Oriented Data Abstraction Layered Architecture Table Driven Interpreters State transition systems
0.25	decrição
0.25	diferenças

3. In the project the server would send every minute the Game Score Board to every client.

3.a. [1.0 points]

- Describe why this type of interaction does not fit into the classical **Client-Server** architectural pattern.

1.0	C-S o servidor só responde a pedidos do clinete pedido-resposta aqui o servidor envia mensagem sem o cliete pedir o cliente nao responde

3.b. [2.0 points]

- With the help of pseudo-code or C code, present how you would implement this functionality as a classical **Client-Server** architectural pattern.
- Present the code for the client and the server.

Client

```
while(1)
  pede_score()
  recebe _score()
```

Server

```
    while(1)
         m = recev_message()
         if (m.tipo = socore_board)
             send_scoreboard()
```

IPC [5.0]

4. [2.0 points]

- Describe how a signal is sent and received in Unix.
- Present all the steps from th+e moment the programmer creates signal to the moment the signal is processed on the other process.

1 – invocation of system call
2 – change of execution to kernel
3 – search for the processed
4 – in the process search for the signal handler
5 – if the signal handler is nor registered process is killed
6 – change execution (user level) to the signal handler

5. [1.0 points]

- Describe the reasons to use the **htons** and **ntohs** functions.
- Present two situations in the project where the two functions should have been used.

In the world there are tow ways to represent 16 bit number big/little endian
The way numbers are represented can be different between two computers commentating and even between a computer and the network layer.
So it is necessary to guarantee that the numbers are always represented in the same way:
htons - converts to the canonical representation
ntohs – converts to the host representations

when defining the port
in the structures sent to other computers

6. [2.0 points]

- Describe what is the objective of the **bind** system call.
- Present and justify the **three** situations where it is obligatory to use the bind.

Bind assigns a port to a socket so that other processes can communication with it
1 – in a INET server socket (to receive connections)
2 – in a DGRAM socket to receive messages
3 – in DGRAM socket so that the receipient of the messaages know the address (for instance to reply)

Processes / Threads / Synchronization [7.0]

7. [1.0 point]

- Explain why threads take less time to be created than processes.

Since threads share some resources with the other threads inside the same process, it is easier for the kernel to create them
Memory allocation, and protection mechanisms do not need to be configured.

8. [2.0 point]

- Explain why **condition variables** need a mutex associated to them.
- Write the pseudo-code of the use of condition variable, and explain what happens to the various variables and threads during the execution of such code.

1 - The condition variable allows a process to get blocked inside a critical region (guarded by such mutex).

2 - When the thread waits on the condition variable the mutex is unlocked.

3 - When such thread is notified, it only continues execution when the mutex is unlocked

thread 1	**thread 2**
lock (m1)	lock (m1)
cond-var_wait(cv, m1)	cond-var_signal(cv)
	unlock(m1)
unlock(m1)	

9. [2.0 points]
Shared memory is the most efficient way to exchange information between two threads.

- Describe one problem that should be handled/solved when using shared memory in various threads.
- Present a simple solution to this problem.
- Exemplify with an example in pseudo-code or C.

On problem is the concurrency to the access of shared data, where race conditions can occur This is solved defining critical regions and guard them using mutexes

int a;	

mutex_lock(m1)
a = a+1
mutex_ unlock(m1)
	mutex_lock(m1)
	a = a-1
	mutex_ unlock(m1)

10. [2.0 points]
Shared memory can be used between different processes in the same machine.

- Present two types of UNIX API that allow the use of shared memory between two processes.
- Present the pseudo-code that allows the generic use of shared memory and describe what each step accomplishes/performs.

SYS V uses numerical identifiers	Memory mapped files uses memory addresses to access files	POSIX uses names to identify the memory regions

• Create memory region (on kernel)	
• Get an address that points to memory region • uses regular pointers to access memory • disconnects from memory region • Get an address that points to memory region • uses regular pointers to access memory • disconnects from memory region	• Get an address that points to memory region • uses regular pointers to access memory • disconnects from memory region
• Destroys memory region	

Operating system organization [2.0]

11. [2.0 point]
The Unix operation system is organized in layers.

- Describe this architecture pattern.
- Present the various layers of the UNIX operating system, describing in detail their responsibilities and interactions.

The system is divided in layers that contain components that share similar responsibilities
each layer only communicates with the immediately lower layer in a client-server fashion

layers:

- applications
- libraries
- system calls
- kernel
- drivers
- HW